DETAILED ACTION

1. Claims 1-16, 41-50, and 66-85 have been examined.

Specification

2. The disclosure is objected to because of the following informalities:

- In paragraph [97], does applicant mean "74_1 - 74_n" instead of "78_1 - 78_n"? The examiner has been unable to find the latter in the drawings.

Appropriate correction is required.

Claim Objections

3. Claim 1 is objected to because of the following informalities: In line 4, please delete the dash between "hardwired" and "pipeline". Appropriate correction is required.

Claim Rejections - 35 USC § 112

4. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

5. Claims 1-6, 9-12, 15-16, 41-42, 44-46, 49-50, 66, 73, 76-77, and 80 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention without undue experimentation. Specifically, in claims 1, 6, 9, 41, 44,

66, 73, 76, and 80, applicant claims "...without generating a virtual address corresponding to the indicated at least one processing pipeline" (and the like). Applicant also makes a distinction in the specification between an address of a pipeline and an instance identifier that identifies the pipeline. See paragraphs [97]-[98]. However, the examiner fails to see the difference between the two, and therefore, it is not understood how a pipeline can be indicated without generating some form of "virtual" address. Applicant has not described what the difference is between an address and an identifier, and to the examiner, they are the same thing with the same purpose: to indicate one of multiple pipelines. Despite the difference in name, an address and an identifier can only be represented in binary form, and therefore, they are both unique sequences of 0s and 1s, where each unique sequence is used to indicate a different pipeline. Hence, it appears as if applicant is trying to exclude pipeline indication by virtual address, which means that an identifier must indicate a pipeline (if neither occurred, then it is not clear how a pipeline can be indicated). However, for the reasons stated above, an identifier is an address.

6. Claims 2-5, 10-12, 15-16, 42, 45-46, 49-50, 77, are rejected under 35 U.S.C. 112, 1st paragraph, for lacking enablement, because they are each dependent on a claim lacking enablement.

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 1-12, 15-16, 41-46, 49-50, 66, 69-70, 73, 76-77, 80, and 83 are rejected under 35 U.S.C. 103(a) as being unpatentable over Inagaki et al., U.S. Patent No. 7,177,310 (herein referred to as Inagaki) in view of the examiner's taking of Official Notice.

9. Referring to claim 1, Inagaki has taught a pipeline accelerator, comprising:

a) a memory. See Fig.1 and note that any combination of components 150-154, 117, 119, and 124, may be considered "a memory".

b) a hardwired circuit coupled to the memory, including processing accelerators (see Fig.1, at least some of components 9 and 11-14), and operable to:

b1) receive a message that includes data and that includes a header having information indicating at least one but fewer than all of the processing accelerators by receiving the data and the information on at least one common bus line. See Fig.1, Fig.3, and column 5, line 65, to column 6, line 17. Packet 134 of Fig.3 is received on the upper bus 1. The destination 163 in the header indicates which of the accelerators will be used to process the message data.

b2) extract the data from the message, load the extracted data into the memory, retrieve the extracted data from the memory, and process the retrieved data with the indicated at least one processing accelerator without first generating a virtual address corresponding to the indicated at least one processing accelerator. See Fig.1 and column 6, lines 29-35, and note that the data to be processed is stored in the appropriate buffer of buffers 150-154. The data is then retrieved from the buffer and processed by the corresponding accelerator of accelerators 9 and 11-14. It should be noted that, although a virtual address corresponding to the indicated at least one processing accelerator is retrieved

from a lookup table (column 6, lines 9-21), the virtual address is not generated, i.e., brought into existence. The virtual address already existed in the table prior to the lookup. It was merely retrieved in response to the lookup. Hence, it can be said that the data is processed without first generating/creating a virtual address.

b3) provide the processed data to an external source. See Fig.1, column 6, lines 61-65, and column 8, lines 62-67. Note that after the data is processed, it is routed to an external location.

b4) Inagaki has not explicitly taught that the circuit is operable to perform the claimed steps without executing a program instruction. However, it should also be noted that Inagaki makes not a single mention of executing instructions to perform the claimed steps. Hence, it is unclear as to whether Inagaki executes instructions to perform the claimed steps. Official Notice is taken that hardware and software are logically equivalent and that anything performed by software can be performed solely by hardware and vice-versa. The choice between hardware and software depends on desired speed, cost, and complexity. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki such that the circuit performs the claimed steps without executing a program instruction, but instead in hardware alone. For example, as is known, an FPGA can be programmed to perform steps in hardware only and without instruction execution.

b5) Inagaki has also not taught that the processing accelerators are pipelined. However, Official Notice is taken that pipelined processing and its advantages are well known and accepted in the art. Specifically, pipelines provide an alternative to slow serial execution

by dividing execution into stages and overlapping stages of different items being processed. This parallel form of execution is akin to an assembly line, which results in increased efficiency and throughput. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki's processing accelerators to be pipelined.

10. Referring to claim 2, Inagaki, as modified, has taught that pipeline accelerator of claim 1. Inagaki has not explicitly taught that the memory is disposed on a first integrated circuit and the pipeline circuit is disposed on a second integrated circuit. However, the separation of components into distinct integrated circuits is not a patentable feature. One of ordinary skill in the art would have recognized that separation of components onto separate chips allows for increased flexibility and reduced repair cost because if just a portion of a system is to be upgraded or repaired, then only that integrated circuit would be upgraded or repaired, as opposed to upgrading or repairing the whole system (if it were on a single chip). As a result, it would have been obvious to modify Inagaki such that the memory 150-154 is implemented on a separate integrated circuit from that of the pipeline circuit 9 and 11-14. Also, see *Nerwin v. Erlichman 168 USPQ 177 (1969)*.

11. Referring to claim 3, Inagaki, as modified, has taught the pipeline accelerator of claim 1. Inagaki has not taught that the pipeline circuit is disposed on a field-programmable gate array. However, Official Notice is taken that a field-programmable gate arrays (FPGA) and its advantages are well known and accepted in the art. Specifically, an FPGA may be efficiently reprogrammed by a designer after manufacture, thereby realizing specialized circuitry for processing in particular environments. The reprogrammability also allows a designer to upgrade,

completely change, or otherwise modify a configuration as needed for increased efficiency. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki such that the pipeline circuit is disposed on a field-programmable gate array.

12. Referring to claim 4, Inagaki, as modified, has taught the pipeline accelerator of claim 1 wherein the pipeline circuit is operable to provide the processed data to the external source by: loading the processed data into the memory; retrieving the processed data from the memory; and providing the retrieved processed data to the external source. See Fig.1 and note that after processing, the processed data is loaded into and retrieved from memory components 119 and 124 and then provided to the external source.

13. Referring to claim 5, Inagaki, as modified, has taught the pipeline accelerator of claim 1, wherein the external source comprises a processor; and the pipeline circuit is operable to receive the data from the processor. Inagaki is concerned with transferring and processing data in a network. Hence, processors are sources and destinations of data.

14. Referring to claim 6, Inagaki has taught a computing machine, comprising:

a) a processor operable to broadcast a message that includes data and that includes a header having information identifying at least one but fewer than all destination accelerators of the data. See Fig.1, and note destination accelerators 9 and 11-14. Also, see column 5, line 65, to column 6, line 17. Packet 134 of Fig.3 is received on the upper bus 1 (inherently from some processor broadcasting an IP packet. Also, component 30 in Fig.1 may be considered a processor). The destination 163 in the header indicates which of the accelerators will be used to process the message data.

b) an accelerator coupled to the processor and comprising:

b1) a memory. See Fig.1 and note that any combination of components 150-154, 117, 119, and 124, may be considered "a memory".

b2) a hardwired circuit coupled to the memory, including at least one processing component (see Fig.1), and operable to:

- receive the message from the processor by receiving the data and the information via at least one same bus line. See Fig.1, Fig.3, and column 5, line 65, to column 6, line 17. Packet 134 of Fig.3 is received on the upper bus 1. The destination 163 in the header indicates which of the accelerators will be used to process the message data.
- extract the data from the message, load the extracted data into the memory, retrieve the extracted data from the memory, and process the retrieved data with the identified at least one destination accelerator without first generating a virtual address corresponding to the identified at least one destination accelerator. See Fig.1 and column 6, lines 29-35, and note that the data to be processed is stored in the appropriate buffer of buffers 150-154. The data is then retrieved from the buffer and processed by the corresponding accelerator of accelerators 9 and 11-14. It should be noted that, although a virtual address corresponding to the indicated at least one destination accelerator is retrieved from a lookup table (column 6, lines 9-21), the virtual address is not generated, i.e., brought into existence. The virtual address already existed in the table prior to the lookup. It was merely retrieved in response to the lookup. Hence, it can be said that the data is processed without first generating/creating a virtual address.

- provide the processed data to the processor. See Fig.1, column 6, lines 61-65, and column 8, lines 62-67. Note that after the data is processed, it is routed back to the processor 30.
- Inagaki has not explicitly taught that the circuit is operable to perform the claimed steps without executing a program instruction. However, it should also be noted that Inagaki makes not a single mention of executing instructions to perform the claimed steps. Hence, it is unclear as to whether Inagaki executes instructions to perform the claimed steps. Official Notice is taken that hardware and software are logically equivalent and that anything performed by software can be performed solely by hardware and vice-versa. The choice between hardware and software depends on desired speed, cost, and complexity. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki such that the circuit performs the claimed steps without executing a program instruction, but instead in hardware alone. For example, as is known, an FPGA can be programmed to perform steps in hardware only and without instruction execution.
- Inagaki has also not taught that at least some of the above components are pipelined. However, Official Notice is taken that pipelined processing and its advantages are well known and accepted in the art. Specifically, pipelines provide an alternative to slow serial execution by dividing execution into stages and overlapping stages of different items being processed. This parallel form of execution is akin to an assembly line, which results in increased efficiency and

throughput. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify at least some of Inagaki's components to be pipelined.

15. Referring to claim 7, Inagaki has taught an accelerator, comprising:

a) a memory. See Fig.1 and note that any combination of components 150-154, 117, 119, and 124, may be considered "a memory".

b) a hardwired circuit (Fig.1) coupled to the memory, and operable to:

b1) receive data without receiving, with the data, information corresponding to a post-processing destination of the data. See Fig.1, Fig.3, and column 5, line 65, to column 6, line 17. Essentially, a packet is received on the upper bus 1 and ultimately processed. The post-processing destination isn't added to the packet until after it is processed. See column 6, line 61, to column 7, line 32.

b2) process the received data, load the processed data into the memory, and retrieve the processed data from the memory. See Fig.1 and note that the data is processed by at least one of processing units 9 and 11-14 before being stored in and ultimately retrieved from memory 119/124.

b3) generate a message header that includes first information indicating a destination of the data, generate a message that includes the processed data and the header, and provide the message to an external source. See Fig.1, column 7, lines 14-32, and column 8, lines 62-67.

b4) Inagaki has not explicitly taught that the circuit is operable to perform the claimed steps without executing a program instruction. However, it should also be noted that

Inagaki makes not a single mention of executing instructions to perform the claimed steps. Hence, it is unclear as to whether Inagaki executes instructions to perform the claimed steps. Official Notice is taken that hardware and software are logically equivalent and that anything performed by software can be performed solely by hardware and vice-versa. The choice between hardware and software depends on desired speed, cost, and complexity. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki such that the circuit performs the claimed steps without executing a program instruction, but instead in hardware alone. For example, as is known, an FPGA can be programmed to perform steps in hardware only and without instruction execution.

b5) Inagaki has also not taught at least some pipelined components. However, Official Notice is taken that pipelined processing and its advantages are well known and accepted in the art. Specifically, pipelines provide an alternative to slow serial execution by dividing execution into stages and overlapping stages of different items being processed. This parallel form of execution is akin to an assembly line, which results in increased efficiency and throughput. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki's accelerator and circuit to be pipelined.

16. Referring to claim 8, Inagaki has taught a computing machine comprising:

a) a processor operable to run at least one software application. See either Fig.2, component 2, or Fig.1, component 30, both of which inherently execute applications (i.e., perform some function). Note that both may collectively be referred to as a processor as well.

b) an accelerator coupled to the processor (see Fig.1) and comprising:

b1) a memory. See Fig.1 and note that any combination of components 150-154, 117, 119, and 124, may be considered "a memory".

b2) a hardwired circuit (Fig.1) coupled to the memory, and operable to:

- receive data from the processor without receiving, with the data, information corresponding to a post-processing destination of the data. See Fig.1, Fig.3, and column 5, line 65, to column 6, line 17. Essentially, a packet is received on the upper bus 1 and ultimately processed. The post-processing destination isn't added to the packet until after it is processed. See column 6, line 61, to column 7, line 32.
- process the received data, load the processed data into the memory, and retrieve the processed data from the memory. See Fig.1 and note that the data is processed by at least one of processing units 9 and 11-14 before being stored in and ultimately retrieved from memory 119/124.
- generate a message header that includes, for the processed data, information that indicates a destination software application running on the processor, generate a message that includes the retrieved processed data and the message header, and provide the message to the processor. See Fig.1, column 7, lines 14-32, and column 8, lines 62-67. The destination is inherently to a processor in a network, and hence to an application running on that processor. Also, the packet's destination is Fig.1, component 30, which is part of the collective processor.

- Inagaki has not explicitly taught that the circuit is operable to perform the claimed steps without executing a program instruction. However, it should also be noted that Inagaki makes not a single mention of executing instructions to perform the claimed steps. Hence, it is unclear as to whether Inagaki executes instructions to perform the claimed steps. Official Notice is taken that hardware and software are logically equivalent and that anything performed by software can be performed solely by hardware and vice-versa. The choice between hardware and software depends on desired speed, cost, and complexity. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki such that the circuit performs the claimed steps without executing a program instruction, but instead in hardware alone. For example, as is known, an FPGA can be programmed to perform steps in hardware only and without instruction execution.
- Inagaki has also not taught that at least some of the above components are pipelined. However, Official Notice is taken that pipelined processing and its advantages are well known and accepted in the art. Specifically, pipelines provide an alternative to slow serial execution by dividing execution into stages and overlapping stages of different items being processed. This parallel form of execution is akin to an assembly line, which results in increased efficiency and throughput. As a result, it would have been obvious to one of

ordinary skill in the art at the time of the invention to modify at least some of Inagaki's components to be pipelined.

17. Referring to claim 9, Inagaki has taught a pipeline accelerator comprising:

a) first and second memories. See 1, and note first memory 150-154 and 117 and second memory 119 and 124.

b) a hardwired circuit (Fig.1) coupled to the first and second memories and comprising:

b1) an input-data handler operable to receive from an external source a first message that includes raw data and that includes a first header having first information specifying at least one destination hardwired processing element, to extract the raw data from the message, and to load the raw data into the first memory without first generating a virtual address corresponding to the at least one destination hardwired processing element specified by the first information. See Fig.1, Fig.3, and column 5, line 65, to column 6, line 17. Packet 134 of Fig.3 is received on the upper bus 1. The destination 163 in the header indicates which of the accelerators will be used to process the message data. Note also that the data is loaded into the first memory before processing (see Fig.1). It should also be noted that, although a virtual address corresponding to the indicated at least one destination processing element is retrieved from a lookup table for each packet received (column 6, lines 9-21), the virtual address is not generated, i.e., brought into existence. The virtual address already existed in the table prior to the lookup. It was merely retrieved in response to the lookup. Hence, it can be said that a virtual address is not generating/creating.

b2) hardwired processing elements including the specified at least one destination hardwired processing element and including at least one other processing element, the specified at least one destination hardwired processing element operable to process data. See Fig.1, component 9 and 11-14.

b3) an interface operable to retrieve the raw data from the first memory, provide the retrieved raw data to the at least one destination hardwired pipeline specified by the first information, and load processed data from the hardwired pipeline into the second memory. See Fig.1 and column 6, line 61, to column 7, line 13.

b4) an output-data handler operable to retrieve the processed data from the second memory, to generate a second header having second information indicating a destination of the processed data, to generate a second message that includes the processed data and the second header, and to provide the second message to the external source by providing the processed data and the second information to the external source via at least one same bus line. See Fig.1, column 7, lines 14-32, and column 8, lines 62-67.

b5) Inagaki has not explicitly taught that the circuit is operable to perform the claimed steps without executing a program instruction. However, it should also be noted that Inagaki makes not a single mention of executing instructions to perform the claimed steps. Hence, it is unclear as to whether Inagaki executes instructions to perform the claimed steps. Official Notice is taken that hardware and software are logically equivalent and that anything performed by software can be performed solely by hardware and vice-versa. The choice between hardware and software depends on desired speed, cost, and complexity. As a result, it would have been obvious to one of ordinary skill in

the art at the time of the invention to modify Inagaki such that the circuit performs the claimed steps without executing a program instruction, but instead in hardware alone. For example, as is known, an FPGA can be programmed to perform steps in hardware only and without instruction execution.

b6) Inagaki has also not taught that at least some of the above components are pipelined. However, Official Notice is taken that pipelined processing and its advantages are well known and accepted in the art. Specifically, pipelines provide an alternative to slow serial execution by dividing execution into stages and overlapping stages of different items being processed. This parallel form of execution is akin to an assembly line, which results in increased efficiency and throughput. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify at least some of Inagaki's components to be pipelined.

18. Referring to claim 10, Inagaki, as modified, has taught a pipeline accelerator as described in claim 9 wherein the first and second memories each include respective first and second ports, the input-data handler is operable to load the raw data via the first port of the first memory, the pipeline interface is operable to retrieve the raw data via the second port of the first memory and to load the processed data via the first port of the second memory, and the output-data handler is operable to retrieve the processed data via the second port of the second memory. See Fig.1.

19. Referring to claim 11, Inagaki, as modified, has taught a pipeline accelerator as described in claim 9. Inagaki has not taught a third memory coupled to the hardwired-pipeline circuit, wherein the hardwired pipeline is operable to generate intermediate data while processing the raw data, and wherein the pipeline interface is operable to load the intermediate data into the

third memory and to retrieve the intermediate data from the third memory. However, Official Notice is taken that using memories of some sort to process data is well known and accepted in the art. For instance, in a CPU (Fig.1, component 9), memories such as a register file and cache are well known and advantageous for providing fast, temporary storage of data during processing. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki to include a third memory as described above.

20. Referring to claim 12, Inagaki, as modified, has taught that pipeline accelerator of claim 9.

a) Inagaki has not taught that the pipeline circuit is disposed on a field-programmable gate array. However, Official Notice is taken that a field-programmable gate arrays (FPGA) and its advantages are well known and accepted in the art. Specifically, an FPGA may be efficiently reprogrammed by a designer after manufacture, thereby realizing specialized circuitry for processing in particular environments. The reprogrammability also allows a designer to upgrade, completely change, or otherwise modify a configuration as needed for increased efficiency. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki such that the pipeline circuit is disposed on a field-programmable gate array.

b) Inagaki has further not taught that the first and second memories are respectively disposed on first and second integrated circuits. However, the separation of components into distinct integrated circuits is not a patentable feature. One of ordinary skill in the art would have recognized that separation of components onto separate chips allows for increased flexibility and reduced repair cost because if just a portion of a system is to be upgraded or repaired, then only that integrated circuit would be upgraded or repaired, as opposed to upgrading or repairing the

whole system (if it were on a single chip). As a result, it would have been obvious to modify Inagaki such that that the first and second memories are respectively disposed on first and second integrated circuits. Also, see *Nerwin v. Erlichman 168 USPQ 177 (1969)*.

21. Referring to claim 15, Inagaki, as modified, has taught a pipeline accelerator as described in claim 9.

a) Inagaki, as modified, has further taught that each of the input-data handler, hardwired pipeline, pipeline interface, and output-data handler has a respective operating configuration. That is, each component in Inagaki operates in a specific fashion and consequently, each component inherently has a respective operating configuration.

b) Inagaki, as modified, has further taught a configuration manager coupled to and operable to set the operating configurations of the input-data handler, hardwired pipeline, pipeline interface, and output-data handler. That is, it is the inherent nature of an FPGA to be coupled to a configuration manager so that the FPGA may be programmed to include the desired functionality.

22. Referring to claim 16, Inagaki, as modified, has taught a pipeline accelerator as described in claim 9.

a) Inagaki, as modified, has inherently taught that each of the input-data handler, hardwired pipeline, pipeline interface, and output-data handler has a respective operating status. That is, at the very least, each component in the system is either operating or not operating (and these are statuses).

b) Inagaki, as modified, has not taught an exception manager coupled to and operable to identify an exception in the input-data handler, hardwired pipeline, pipeline interface, or output-data

handler in response to the operating statuses. However, Official Notice is taken that checking for errors in a pipeline during processing is well known and accepted in the art. Any type of error which causes an exception should be monitored so that the system can take appropriate action to correct the error. As a result, in order to ensure proper execution, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki such that an exception manager is implemented to detect exceptions.

23. Referring to claim 41, the method of claim 41 is performed by the circuit of claim 1. Consequently, claim 41 is rejected for the same reasons set forth in the rejection of claim 1. Also, though Inagaki has not taught that the header has information indicating a size of the message, the examiner asserts that this concept is well known and accepted in the art. By indicating the size of the message, it is clear how much data is to be received/processed. This is useful in at least systems that handle variable length packets. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki to indicate a size of the message within the header.

24. Referring to claim 42, Inagaki, as modified, has taught a method as described in claim 41. Furthermore, the method of claim 42 is performed by the circuit of claim 4. Consequently, claim 42 is rejected for the same reasons set forth in the rejection of claim 4.

25. Referring to claim 43, the method of claim 43 is performed by the accelerator of claim 7. Consequently, claim 43 is rejected for the same reasons set forth in the rejection of claim 7. In addition, the providing the message to an external source comprises providing on a single bus. See Fig.1 and note that the processed data is outputted via a single bus.

26. Referring to claim 44, the method of claim 44 is performed by the circuit of claim 9. Consequently, claim 44 is rejected for the same reasons set forth in the rejection of claim 9.

27. Referring to claim 45, Inagaki, as modified, has taught a method as described in claim 44. Furthermore, the method of claim 45 is performed by the circuit of claim 10. Consequently, claim 45 is rejected for the same reasons set forth in the rejection of claim 10.

28. Referring to claim 46, Inagaki, as modified, has taught a method as described in claim 44. Furthermore, the method of claim 46 is performed by the circuit of claim 11. Consequently, claim 46 is rejected for the same reasons set forth in the rejection of claim 11.

29. Referring to claim 49, Inagaki, as modified, has taught a method as described in claim 44, further comprising setting parameters for loading and retrieving the raw data, processing the retrieved data, and loading and providing the processed data. Recall from Fig.1 that data is sent to buffers before/after processing. Hence, pointers parameters need to be set to load and store data from/to buffers/queues, etc.

30. Referring to claim 50, Inagaki, as modified, has taught a method as described in claim 44. While Inagaki has not explicitly taught determining whether an error occurs during the loading and retrieving of the raw data, the processing of the retrieved data, and the loading and providing of the processed data, Official Notice is taken that checking for errors during processing is well known and accepted in the art. Any type of error, which causes an exception, should be monitored so that the system can take appropriate action to correct the error. As a result, in order to ensure proper execution, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki to determining whether an error occurs during the

loading and retrieving of the raw data, the processing of the retrieved data, and the loading and providing of the processed data.

31. Referring to claim 66, Inagaki has taught a pipeline accelerator comprising:

a) a memory. See Fig.1 and note that any combination of components 150-154, 117, 119, and 124, may be considered "a memory".

b) a hardwired circuit coupled to the memory, including at least one processing accelerator (see Fig.1, at least one of components 9 and 11-14), and operable to:

b1) receive a message that includes data and that includes a header having information indicating at least one but not all processing accelerators disposed in the hardwired circuit by receiving the data and the information on at least one common bus line. See Fig.1, Fig.3, and column 5, line 65, to column 6, line 17. Packet 134 of Fig.3 is received on the upper bus 1. The destination 163 in the header indicates which of the accelerators will be used to process the message data.

b2) extract the data from the message, load the extracted data into the memory, retrieve the extracted data from the memory, and process the retrieved data with the at least one processing accelerator specified by the information. See Fig.1 and column 6, lines 29-35, and note that the data to be processed is stored in the appropriate buffer of buffers 150-154. The data is then retrieved from the buffer and processed by the corresponding accelerator of accelerators 9 and 11-14.

b3) provide the processed data to an external source. See Fig.1, column 6, lines 61-65, and column 8, lines 62-67. Note that after the data is processed, it is routed to an external location.

b4) to extract from the header the information specifying the at least one processing accelerator. See Fig.3, component 163, and column 6, lines 9-13.

b5) to generate from the extracted information an identifier other than a virtual address that identifies the specified processing accelerator. See column 6, lines 9-21. The header information is used to derive a MAC address that identifies the processing accelerator. It should also be noted that, although a virtual MAC address corresponding to specified processing accelerator is retrieved from a lookup table for each packet received (column 6, lines 9-21), the virtual address is not generated, i.e., brought into existence. The virtual address already existed in the table prior to the lookup. It was merely retrieved in response to the lookup. Hence, it can be said that a virtual address is not generating/creating.

b6) to store the identifier in association with the data. See Fig.3, component 130. The data and MAC address 160 are stored together in memory (see column 6, lines 22-35).

b7) to provide the retrieved data to the pipeline in response to the stored identifier. See Fig.1. From memory, the data is processed.

b8) Inagaki has also not taught that at least the processing accelerators are pipelined. However, Official Notice is taken that pipelined processing and its advantages are well known and accepted in the art. Specifically, pipelines provide an alternative to slow serial execution by dividing execution into stages and overlapping stages of different items being processed. This parallel form of execution is akin to an assembly line, which results in increased efficiency and throughput. As a result, it would have been obvious to

one of ordinary skill in the art at the time of the invention to modify Inagaki's processing accelerators to be pipelined.

b9) Inagaki has not taught that the processing of the retrieved data is performed without executing a program instruction. However, it should also be noted that Inagaki makes not a single mention of executing instructions to perform the claimed steps. Hence, it is unclear as to whether Inagaki executes instructions to perform the claimed steps. Official Notice is taken that hardware and software are logically equivalent and that anything performed by software can be performed solely by hardware and vice-versa. The choice between hardware and software depends on desired speed, cost, and complexity. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki such that the circuit performs the processing step without executing a program instruction, but instead in hardware alone. For example, as is known, an FPGA can be programmed to perform steps in hardware only and without instruction execution.

32. Referring to claim 69, Inagaki, as modified, has taught the pipeline accelerator as described in claim 7, wherein the hardwired-pipeline circuit is further operable to:

a) store in association with the processed data second information indicating the destination of the processed data. See Fig.3, component 130. The data and MAC address 160 (second information) are stored together in memory (see column 6, lines 22-35)

b) generate the message header in response to the second information. See Fig.3 and column 6, lines 9-21. An Ethernet header is generated in response to the MAC address.

33. Referring to claim 70, Inagaki, as modified, has taught a pipeline accelerator as described in claim 69 wherein the second information equals the first information. See Fig.3 and column 6, lines 9-35. Basically, a destination IP (first information) is mapped to a particular MAC address (second information), which in turn identifies the pipeline to perform the processing. Because they are mapped 1-to-1, they are equal. That is, both the IP and MAC identify the pipeline.

34. Referring to claim 73, claim 73 is rejected for the same reasons set forth in the rejections of claims 9 and 66.

35. Referring to claim 76, claims 76 is rejected for reasons set forth in rejections of previous independent claims (most notably, but not limited to, 9 and 73). Also, Inagaki has taught wherein the pipeline interface is further operable, without executing a program instruction, to store in association with the processed data second information indicating the destination of the processed data, and wherein the output-data handler is further operable, without executing a program instruction, to generate the first information from the second information. See column 7, lines 14-32, and note that second information (IP address) is used to generate first information through route table lookup.

36. Referring to claim 77, Inagaki, as modified, has taught the pipeline accelerator of claim 76 wherein the second information equals the first information. See column 7, lines 14-32. Basically, IP information (second information) is mapped to particular routine module information (first information), which in turn identifies the destination of the processed packet. Because the IP and routine module information are mapped 1-to-1, they are equal. That is, both the IP and routine module information identify a destination.

37. Referring to claim 80, claim 80 is rejected for the same reasons set forth in the rejection of claim 66.

38. Referring to claim 83, Inagaki, as modified, has taught a method as described in claim 43. Furthermore, claim 83 is rejected for the same reasons set forth in the rejection of 69.

39. Claims 13-14, 47-48, 67-68, 71-72, 74-75, 78-79, 81-82, and 84-85 are rejected under 35 U.S.C. 103(a) as being unpatentable over Inagaki in view of the examiner's taking of Official Notice and further in view of Bishop et al., U.S. Patent No. 4,914,653 (herein referred to as Bishop).

40. Referring to claim 13, Inagaki, has taught an accelerator comprising:

a) first and second memories. See 1, and note first memory 150-154 and 117 and second memory 119 and 124.

b) a hardwired circuit (Fig.1) coupled to the first and second memories and comprising:

b1) an input-data handler operable to receive from an external source a first message that includes raw data and that includes a first header having first information specifying at least one destination hardwired processing element, to extract the raw data from the message, and to load the raw data into the first memory without first generating a virtual address corresponding to the at least one destination hardwired processing element specified by the first information. See Fig.1, Fig.3, and column 5, line 65, to column 6, line 17. Packet 134 of Fig.3 is received on the upper bus 1. The destination 163 in the header indicates which of the accelerators will be used to process the message data. Note also that the data is loaded into the first memory before processing (see Fig.1). It should

also be noted that, although a virtual address corresponding to the indicated at least one destination processing element is retrieved from a lookup table for each packet received (column 6, lines 9-21), the virtual address is not generated, i.e., brought into existence. The virtual address already existed in the table prior to the lookup. It was merely retrieved in response to the lookup. Hence, it can be said that a virtual address is not generating/creating.

b2) hardwired processing elements including the specified at least one destination hardwired processing element and including at least one other processing element, the specified at least one destination hardwired processing element operable to process data. See Fig.1, component 9 and 11-14.

b3) an interface operable to retrieve the raw data from the first memory, provide the retrieved raw data to the at least one destination hardwired pipeline specified by the first information, and load processed data from the hardwired pipeline into the second memory. See Fig.1 and column 6, line 61, to column 7, line 13.

b4) an output-data handler operable to retrieve the processed data from the second memory, to generate a second header having second information indicating a destination of the processed data, to generate a second message that includes the processed data and the second header, and to provide the second message to the external source by providing the processed data and the second information to the external source via at least one same bus line. See Fig.1, column 7, lines 14-32, and column 8, lines 62-67.

b5) Inagaki has not explicitly taught that the circuit is operable to perform the claimed steps without executing a program instruction. However, it should also be noted that

Inagaki makes not a single mention of executing instructions to perform the claimed steps. Hence, it is unclear as to whether Inagaki executes instructions to perform the claimed steps. Official Notice is taken that hardware and software are logically equivalent and that anything performed by software can be performed solely by hardware and vice-versa. The choice between hardware and software depends on desired speed, cost, and complexity. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki such that the circuit performs the claimed steps without executing a program instruction, but instead in hardware alone. For example, as is known, an FPGA can be programmed to perform steps in hardware only and without instruction execution.

b6) Inagaki has also not taught that at least some of the above components are pipelined. However, Official Notice is taken that pipelined processing and its advantages are well known and accepted in the art. Specifically, pipelines provide an alternative to slow serial execution by dividing execution into stages and overlapping stages of different items being processed. This parallel form of execution is akin to an assembly line, which results in increased efficiency and throughput. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify at least some of Inagaki's components to be pipelined.

c) Inagaki has also not taught an input-data queue coupled to the input-data handler and the pipeline interface, wherein the input-data handler is operable to load into the input-data queue a pointer to a location of the raw data within the first memory and wherein the pipeline interface is operable to retrieve the raw data from the location using the pointer. However, Bishop has

taught the concept of queuing pointers to input data in order to reduce latency during transmission. See column 2, line 62, to column 3, line 5. Therefore, in order efficiently transmit packets, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki to include a multi-packet input buffer and an input-data queue for holding a pointer to a packet (raw data) within the buffer and then using the pointer to locate the desired packet.

41. Referring to claim 14, claim 14 is largely rejected for the same reasons set forth in the rejection of claim 10. Furthermore, Inagaki has not taught an output data queue coupled to the output-data handler and the pipeline interface, wherein the pipeline interface is operable to load into the output-data queue a pointer to a location of the processed data within the second memory, and wherein the output-data handler is operable to retrieve the processed data from the location using the pointer. However, Bishop has taught the concept of queuing pointers to output data in order to reduce latency during transmission. See column 2, line 62, to column 3, line 5. Therefore, in order efficiently transmit packets, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki to include an output data queue coupled to the output-data handler and the pipeline interface, wherein the pipeline interface is operable to load into the output-data queue a pointer to a location of the processed data within the second memory, and wherein the output-data handler is operable to retrieve the processed data from the location using the pointer.

42. Referring to claim 47, Inagaki, as modified, has taught a method as described in claim 44. Furthermore, the method of claim 47 is performed by the circuit of claim 13. Consequently, claim 47 is rejected for the same reasons set forth in the rejection of claim 13.

43. Referring to claim 48, claim 48 is largely rejected for the same reasons set forth in the rejection of claim 9. Furthermore, Inagaki has not taught loading into an output message queue a pointer to a location of the processed data within the second memory, and wherein retrieving the processed data comprises retrieving the processed data from the location using the pointer. However, Bishop has taught the concept of queuing pointers to output data in order to reduce latency during transmission. See column 2, line 62, to column 3, line 5. Therefore, in order efficiently transmit packets, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki to perform loading into an output message queue a pointer to a location of the processed data within the second memory, and wherein retrieving the processed data comprises retrieving the processed data from the location using the pointer.

44. Referring to claim 67, claim 67 is largely rejected for the same reasons set forth in the rejection of claim 66.

a) Inagaki has further taught to extract from the header the information indicating the destination of the data (see Fig.3, component 163, and column 6, lines 9-13), to generate from the extracted information an identifier that identifies the processing accelerator corresponding to the destination (see column 6, lines 9-21. The header information is used to derive a MAC address that identifies the processing accelerator), and to store the identifier in association with the data (see Fig.3, component 130. The data and MAC address 160 are stored together in memory (see column 6, lines 22-35)).

b) Inagaki has not taught to store a pointer to the extracted data. However, Bishop has taught the concept of storing/queuing a pointer to input data in order to reduce latency during transmission. See column 2, line 62, to column 3, line 5. Therefore, in order efficiently transmit packets, it

would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki to store a pointer to the extracted data.

c) Inagaki, as modified, has further taught to provide the retrieved data to the pipeline in response to the stored identifier. This is inherent given parts (a)-(b) above. Since the MAC address and data are needed to determine which data is processed by which unit, then the providing occurs in response to the identifier and pointer.

45. Referring to claim 68, Inagaki has taught an accelerator, comprising:

a) a memory. See Fig.1 and note that any combination of components 150-154, 117, 119, and 124, may be considered "a memory".

b) a hardwired circuit coupled to the memory, including at least one processing accelerator (see Fig.1, at least one of components 9 and 11-14), and operable to:

b1) receive a message that includes data and that includes a header having information uniquely identifying each of at least one pipeline. See Fig.1, Fig.3, and column 5, line 65, to column 6, line 17. Packet 134 of Fig.3 is received on the upper bus 1. The destination 163 in the header indicates which of the accelerators will be used to process the message data.

b2) extract the data from the message, load the extracted data into the memory, retrieve the extracted data from the memory, and process the retrieved data with the at least one accelerator identified by the information. See Fig.1 and column 6, lines 29-35, and note that the data to be processed is stored in the appropriate buffer of buffers 150-154. The data is then retrieved from the buffer and processed by the corresponding accelerator of accelerators 9 and 11-14.

b3) provide the processed data to an external source. See Fig.1, column 6, lines 61-65, and column 8, lines 62-67. Note that after the data is processed, it is routed to an external location.

b4) extract from the header the information. See column 6, lines 9-13.

b5) Inagaki has not explicitly taught performing the claimed steps without executing a program instruction. However, it should also be noted that Inagaki makes not a single mention of executing instructions to perform the claimed steps. Hence, it is unclear as to whether Inagaki executes instructions to perform the claimed steps. Official Notice is taken that hardware and software are logically equivalent and that anything performed by software can be performed solely by hardware and vice-versa. The choice between hardware and software depends on desired speed, cost, and complexity. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki such that the circuit performs the claimed steps without executing a program instruction, but instead in hardware alone. For example, as is known, an FPGA can be programmed to perform steps in hardware only and without instruction execution.

b6) Inagaki has also not taught that at least the processing accelerators are pipelined. However, Official Notice is taken that pipelined processing and its advantages are well known and accepted in the art. Specifically, pipelines provide an alternative to slow serial execution by dividing execution into stages and overlapping stages of different items being processed. This parallel form of execution is akin to an assembly line, which results in increased efficiency and throughput. As a result, it would have been obvious to

one of ordinary skill in the art at the time of the invention to modify Inagaki's processing accelerators to be pipelined.

b7) Inagaki has further not taught storing a pointer to the extracted data in a location associated with the pipeline corresponding to the destination. However, Bishop has taught the concept of queuing pointers to input data in order to reduce latency during transmission. See column 2, line 62, to column 3, line 5. Therefore, in order efficiently transmit packets, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki to include a multi-packet input buffer and an input-data queue for holding a pointer to a packet (raw data) within the buffer and then using the pointer to locate the desired packet.

b8) Inagaki, as modified, has further taught providing the retrieved data to the pipeline in response to the stored pointer. Clearly, in view of b7 above, such a limitation is inherent.

46. Referring to claim 71, Inagaki has taught a pipeline accelerator, comprising:

a) a memory. See Fig.1 and note that any combination of components 150-154, 117, 119, and 124, may be considered "a memory".

b) a hardwired circuit coupled to the memory (see Fig.1), and operable to:

b1) receive data without receiving, with the data, information corresponding to a post-processing destination of the data. See Fig.1, Fig.3, and column 5, line 65, to column 6, line 17. Essentially, a packet is received on the upper bus 1 and ultimately processed. The post-processing destination isn't added to the packet until after it is processed. See column 6, line 61, to column 7, line 32.

b2) process the received data, load the processed data into the memory, and retrieve the processed data from the memory. See Fig.1 and note that the data is processed by at least one of processing units 9 and 11-14 before being stored in and ultimately retrieved from memory 119/124.

b3) generate a message header that includes first information indicating a destination of the processed data, generating a message that includes the processed data and the header, and then providing the message to the external source. See Fig.1, column 7, lines 14-32, and column 8, lines 62-67.

b4) Inagaki has not taught that at least some components are pipelined. However, Official Notice is taken that pipelined processing and its advantages are well known and accepted in the art. Specifically, pipelines provide an alternative to slow serial execution by dividing execution into stages and overlapping stages of different items being processed. This parallel form of execution is akin to an assembly line, which results in increased efficiency and throughput. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki's processing components to be pipelined.

b5) Inagaki has also not taught that claimed steps are performed without executing a program instruction. However, Official Notice is taken that hardware and software are logically equivalent and that anything performed by software can be performed solely by hardware and vice-versa. The choice between hardware and software depends on desired speed, cost, and complexity. As a result, it would have been obvious to one of ordinary

skill in the art at the time of the invention to modify Inagaki such that the claimed steps are performed without executing a program instruction, but instead in hardware alone.

b6) Inagaki has not taught storing a pointer to the processed data, storing in association with the pointer second information indicating the destination of the processed data, retrieving the processed data in response to the pointer, and generating the message header in response to the second information. However, Bishop has taught the concept of queuing pointers to output data in order to reduce latency during transmission. See column 2, line 62, to column 3, line 5. Therefore, in order efficiently transmit packets, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Inagaki to store a pointer to the processed data, and retrieving the processed data in response to the pointer.

b7) Inagaki has further taught storing in association with the pointer second information indicating the destination of the processed data and generating the message header in response to the second information. See Fig.3 and column 7, lines 14-32. Note that the IP destination is again stored and used to generate the header.

47. Referring to claim 72, claim 72 is largely rejected for the same reasons set forth in the rejection of claim 71. Furthermore, Inagaki has not taught to store a pointer to the processed data in a location with the destination of the processed data, to retrieve the processed data in response to the pointer, and to generate the message header in response to the location. However, Bishop has taught the concept of queuing pointers to output data in order to reduce latency during transmission. See column 2, line 62, to column 3, line 5. Therefore, in order efficiently transmit packets, it would have been obvious to one of ordinary skill in the art at the

time of the invention to modify Inagaki to store a pointer to the processed data in a location with the destination of the processed data, to retrieve the processed data in response to the pointer, and to generate the message header in response to the location.

48. Referring to claims 74-75, claims 74-75 are rejected for reasons set forth in rejections of previous independent claims (most notably, but not limited to, 73 and 13).

49. Referring to claim 78, claim 78 is rejected for reasons set forth in rejections of previous independent claims (most notably, but not limited to, 76 and 14).

50. Referring to claim 79, claim 79 is rejected for reasons set forth in rejections of previous independent claims (most notably, but not limited to, 76 and 14).

51. Referring to claim 81, claim 81 is rejected for reasons set forth in rejections of previous independent claims (most notably, but not limited to, 80 and 13).

52. Referring to claim 82, claim 82 is rejected for reasons set forth in rejections of previous independent claims (most notably, but not limited to, 48 and 14).

53. Referring to claim 84, the method of claim 84 is performed by the accelerator of claim 71. Consequently, claim 84 is rejected for the same reasons set forth in the rejection of claim 71.

54. Referring to claim 85, the method of claim 85 is performed by the accelerator of claim 72. Consequently, claim 85 is rejected for the same reasons set forth in the rejection of claim 72.

Response to Arguments

55. Applicant's arguments filed on January 6, 2010, have been fully considered but they are not persuasive.

56. Applicant argues the novelty/rejection of claim 1 on page 29 of the remarks, in substance that:

> (1) "In contrast, Inagaki does not teach or suggest processing data with a pipeline without first generating a virtual address corresponding to the pipeline."
>
> (2) "Under the Examiner's premise, no apparatus that performs in hardware what was previously performed in software can ever be patentable."

57. These arguments are not found persuasive for the following reasons:

a) Regarding the first argument, per dictionary.com, definitions of "generate" include "to bring into existence" and "to create…". Based on these definitions, the examiner asserts that Inagaki does not generate a virtual address before processing data. Specifically, a data packet's IP address is stripped and used to lookup a corresponding virtual address before processing. However, the virtual address is not generated, i.e., it's not created or brought into existence. Virtual addresses already exist in a lookup table just prior to processing. Hence, the examiner feels that the current claim language is not narrow enough to overcome Inagaki.

b) Regarding the second argument, minus some unexpected result of implementing a concept in software instead of hardware (or vice-versa), the examiner asserts that one way is not patentable over the other, as hardware and software are logically equivalent. Logic implemented in hardware can also be implemented in software, and vice-versa.

58. Applicant argues the novelty/rejection of claim 7 on page 31 of the remarks, in substance that:

> "In contrast, Inagaki does not teach or suggest a circuit operable to receive data without receiving, with the data, information corresponding to a post-processing destination of the data. Referring to FIGS. 1 and 3 and columns 6 and 7, Inagaki's module 6 receives an IP packet 134 that includes a destination IP address 163, which indicates a post-processing destination of the packet (e.g., col. 7, lines 19-32). The destination IP address 163 stays with the packet IP 134 throughout the packet's journey through the module 6 (e.g., col. 7, lines 19-21). If the module 6

> did not receive the destination IP address 163 with the IP packet 134, then the module would be unable to route the processed packet 134 to its proper external destination because the module would not "know" where to send the processed packet."

59. These arguments are not found persuasive for the following reasons:

a) From column 7, lines 14-32, information pertaining to a routing module to which the processed data is to be transmitted isn't obtained until after data processing. The route table information, which is information corresponding to a post-processing destination of the data, is not received with the data before processing, but after processing when the route information table is consulted. Hence, the data is received without receiving, with the data, the route information.

60. Applicant argues the novelty/rejection of claim 13 on page 36 of the remarks, in substance that:

> "In contrast, referring to FIG. 1, column 6, line 30 - column 7, line 13, and column 7, lines 33-63, Inagaki discloses the module 6 having separate memories 151- 154, separate transmitting buffers 114-117, and separate receiving buffers 120-123, which each correspond to a respective one of the function accelerators 11-14.
>
> Because Inagaki's module 6 includes a separate memory, a separate transmitting buffer, and a separate and receiving buffer for each function accelerator, the module 6 does not need queues that store pointers. In fact, adding such queues to Inagaki's module 6 would at best be redundant, and would increase the complexity of the module 6 with no apparent benefit to offset the increased complexity.
>
> Furthermore, not only would it have been redundant to modify Inagaki's module 6 to include queues that store pointers, but Inagaki teaches away from modifying the module 6 in such a manner. In, e.g., column 7, lines 33-63, and column 8, lines 16-35, Inagaki stresses the importance of maintaining separate memories 151- 154, separate transmitting buffers 114-117, and separate receiving buffers 120-123, because this provides advantages that a common memory and common buffers using queues and pointers cannot provide. For example, referring to column 8, lines 30-35, Inagaki states that using separate memories 151-154 and transmitting buffers 114-117 allows these buffers to be clocked at different speeds; this would be difficult, if not impossible, if the memories and transmitting buffers were not separate, but were implemented as different sections of a same memory."

61. These arguments are not found persuasive for the following reasons:

a) As taught by the prior art, the packet data itself can be buffered and routed through the system (Inagaki), or the packet data can be stored in a memory, and a pointer to the packet data can be buffered and routed through the system until the packet is actually required (Bishop). The latter, as Bishop states in column 3, lines 3-5, reduces latency because pointers are smaller than packet data and therefore, less data needs to be buffered/routed through the system between protocol layers. Applicant notes that it is not obvious to modify Inagaki in view of Bishop because of the advantages realized in Inagaki from having dedicated, asynchronous processing buffers. However, the examiner is not proposing that such buffers be removed. The examiner is merely proposing that instead of routing the full packet data through each component in Inagaki, just a pointer is, and when the packet data is actually needed, the pointer may be used to retrieve it. In fact, Bishop even supports dedicated, asynchronous pointer queues. See column 2, lines 44-66. Hence, each queue in Inagaki, much like those in Bishop, may still be dedicated to a particular accelerator while still being asynchronous with respect to the other queues. The modification, however, lies in what is actually stored in the queues. In Inagaki, it's the full amount of packet data. In Bishop, it's a smaller pointer value, which are quicker to transmit and require less bus lines to do so.

62. All other arguments are deemed not persuasive for the same reasons the above arguments were deemed non-persuasive.

Conclusion

63. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to DAVID J. HUISMAN whose telephone number is (571)272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/David J. Huisman/
Primary Examiner, Art Unit 2183